МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

МОСКОВСКИЙ ПОЛИТЕХ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

ОТЧЕТ ПО ВАРИАТИВНОЙ ЧАСТИ ЗАДАНИЯ

На тему:

«Анализ методик тестирования защищенности веб-приложений»

Выполнили: ст. гр. 241-371 Меркулов Г. С, Ермаков М. А.

Руководитель: Шорников А. В.

Место проведения: Московский Политех, лаборатория «Программно-аппаратных средств обеспечения информационной безопасности»

Москва – 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность тестирования защищенности веб-приложений С развитием цифровых технологий веб-приложения стали ключевым элементом бизнеса, государственных услуг и социальных коммуникаций. Однако рост их распространенности сопровождается увеличением количества кибератак. По данным OWASP Top 10 (2023), наиболее критичными угрозами остаются инъекции, аутентификация с уязвимостями и небезопасные настройки безопасности. Ежегодные отчеты Verizon DBIR и Positive Technologies подтверждают, что более 70% успешных атак происходят из-за недостатков в защите веб-приложений.

Основные угрозы и уязвимости Современные угрозы включают широкий спектр уязвимостей: от SQL-инъекций и XSS до небезопасного управления сессиями и уязвимостей конфигурации. OWASP Top 10 выделяет наиболее опасные категории, которые служат ориентиром для тестирования. Кроме того, Verizon DBIR (2023) указывает, что основной вектор атак — эксплуатация уязвимостей в веб-приложениях, часто вследствие отсутствия должного тестирования.

Цели и задачи анализа методик тестирования безопасности Целью данной работы является всесторонний анализ существующих методик тестирования защищенности веб-приложений, оценка их эффективности, применимости и ограничений. Основные задачи: Классификация существующих методов тестирования;Обзор основных уязвимостей и методов их выявления; Сравнение инструментов и подходов; Анализ методологий и стандартов; Изучение практических аспектов тестирования; Выявление перспективных направлений развития.

ГЛАВА 1. Основные уязвимости и методы их выявления

Инъекции (SQLi, XSS, Command Injection) Инъекционные уязвимости возникают при недостаточной фильтрации входных данных. SQL-инъекции позволяют атакующему выполнять произвольные SQL-запросы к базе данных, получая несанкционированный доступ. Для выявления используются как ручные методы (инъекции через поля ввода), так и автоматизированные инструменты (SQLmap, Burp Suite Intruder).

XSS-атаки делятся на отражённые, хранимые и DOM-based. Они позволяют внедрить вредоносный JavaScript-код, который исполняется у жертвы. Для тестирования применяются такие инструменты, как XSSer, а также модуль Active Scan в Burp Suite.

Command Injection представляет собой внедрение команд ОС через уязвимые формы или параметры. Тестируется путём ввода специальных символов (например, `;`, `&&`) и анализа отклика сервера.

Небезопасная аутентификация и управление сессиями Уязвимости в механизмах аутентификации включают: возможность перебора паролей (brute force), перехват токенов, отсутствие многофакторной аутентификации. Проверка включает подбор слабых паролей, тестирование входа без подтверждения, проверку на устаревшие механизмы сессий (например, session ID в URL).

Недостатки контроля доступа (IDOR, Privilege Escalation) IDOR (Insecure Direct Object Reference) возникает при отсутствии проверки прав доступа к объектам по их идентификаторам. Тестирование включает попытки доступа к ресурсам других пользователей через подмену ID. Эскалация привилегий может быть обнаружена при попытке выполнения действий от имени администратора с обычной учетной записи.

Уязвимости конфигурации (CORS, Headers, SSL/TLS) Ошибки в конфигурации серверов часто становятся точками входа для атак. Тестирование включает анализ HTTP-заголовков (например, `X-Content-Type-Options`, `Content-Security-Policy`), проверку на открытые CORS-политики, использование устаревших или слабых шифров SSL/TLS.

Компоненты с известными уязвимостями (SCA-анализ) Использование библиотек с известными уязвимостями угрожает всей системе. Применяется Software Composition

Analysis (SCA), с помощью таких инструментов, как OWASP Dependency-Check, Snyk, Retire.js. Они анализируют зависимости и сообщают о CVE уязвимостях.

ГЛАВА 2. Инструменты тестирования защищенности

Автоматизированные сканеры (Burp Suite, OWASP ZAP, Nessus) Автоматизированные сканеры обеспечивают быстрое выявление распространённых уязвимостей. OWASP ZAP — бесплатный инструмент с широкими возможностями для анализа веб-приложений, включая сканирование, интерактивный прокси и отчетность. Burp Suite — один из наиболее мощных инструментов, включающий множество модулей: Intruder, Repeater, Scanner (в Pro-версии). Nessus применяется для анализа уязвимостей на уровне операционной системы, сетевых служб и приложений.

Фреймворки для пентеста (Metasploit, SQLmap, BeEF) Metasploit Framework — мощный инструмент для эксплуатации уязвимостей, тестирования защиты и написания собственных эксплойтов. SQLmap — специализированный инструмент для проведения атак SQL-инъекций, автоматизирующий процесс от поиска до извлечения данных. BeEF (Browser Exploitation Framework) ориентирован на эксплуатацию XSS и уязвимостей браузеров, используется в связке с социальной инженерией.

Статические анализаторы (SonarQube, Checkmarx, Semgrep) Эти инструменты анализируют исходный код и конфигурационные файлы на наличие ошибок и потенциальных уязвимостей. SonarQube позволяет интеграцию в CI/CD, поддерживает множество языков. Checkmarx применяется в крупных организациях, обладает высокой точностью. Semgrep выделяется гибкостью настройки и расширяемостью.

Средства анализа зависимостей (Dependency-Check, Snyk) Dependency-Check (от OWASP) проверяет зависимости на наличие уязвимостей по базе данных CVE. Snyk — облачный инструмент, обеспечивающий непрерывный мониторинг зависимостей, с поддержкой языков JavaScript, Python, Java и др. Также применяются такие решения, как GitHub Dependabot и Retire.js.

Сравнение возможностей и эффективности инструментов Выбор инструментов зависит от целей тестирования и этапа SDLC. SAST-инструменты лучше подходят на ранних этапах разработки, в то время как DAST и IAST эффективны при тестировании

готовых решений. Ручные инструменты, такие как Burp Suite и Metasploit, наиболее полезны при пентестах, когда требуется гибкий анализ. Комбинирование различных решений обеспечивает наибольшую полноту картины и снижает риск пропуска критических уязвимостей.

ГЛАВА 3. Методологии и стандарты тестирования

OWASP Testing Guide OWASP Testing Guide (v4) представляет собой структурированную методологию тестирования безопасности веб-приложений. Он охватывает весь жизненный цикл тестирования — от сбора информации до анализа логики приложения. Руководство содержит десятки конкретных проверок, сгруппированных по категориям: аутентификация, управление сессиями, авторизация, ввод данных, ошибки конфигурации и др. Используется как основа для составления чек-листов и обучения специалистов.

PTES (Penetration Testing Execution Standard) PTES представляет собой стандарт, описывающий пошаговое проведение тестов на проникновение. Он включает фазы: предварительная подготовка, сбор информации, моделирование угроз, уязвимости, эксплуатация, постэксплуатация и отчетность. PTES полезен при организации пентестов в крупных организациях, так как охватывает как технические, так и организационные аспекты.

NIST SP 800-115 Руководство NIST SP 800-115 «Technical Guide to Information Security Testing and Assessment» ориентировано на государственные и корпоративные структуры. Оно предлагает систематизированный подход к планированию, выполнению и документированию тестов. Стандарт включает тесты на уязвимости, сканирование, анализ конфигурации, социальную инженерию и физические проверки.

ISO/IEC 27001 и 27034 Международные стандарты ISO/IEC 27001 и 27034 ориентированы на управление информационной безопасностью и безопасность приложений соответственно. ISO/IEC 27001 описывает процесс построения и поддержки системы управления информационной безопасностью (СУИБ), включая элементы аудита и оценки рисков. ISO/IEC 27034 дополняет его, фокусируясь на безопасной разработке и встраивании процессов тестирования в SDLC. Использование этих стандартов обеспечивает соответствие требованиям регуляторов и повышает доверие со стороны клиентов и партнёров.

ГЛАВА 4. Практические аспекты тестирования

Планирование тестирования (Scope, Rules of Engagement) Процесс тестирования начинается с четкого определения объема (scope) и согласования правил взаимодействия (rules of engagement). Определение границ (IP-диапазоны, приложения, API и т.п.) критично для правовой легитимности теста. На этом этапе также уточняются временные рамки, допустимые методы, сценарии остановки и порядок эскалации инцидентов.

Этапы тестирования (Reconnaissance, Scanning, Exploitation, Reporting)

Reconnaissance (разведка): сбор общедоступной информации об объекте (OSINT), включая домены, субдомены, утечки, email-адреса.

Scanning: автоматизированный и ручной анализ портов, сервисов, директорий, уязвимостей.

Exploitation: попытка эксплуатации уязвимостей для подтверждения их наличия и оценки воздействия.

Reporting: составление отчета с подробным описанием обнаруженных проблем, их рисков, рекомендаций по устранению.

Особенности тестирования API (REST, GraphQL) API-интерфейсы часто становятся целями атак, особенно в микросервисных архитектурах. Для REST API проверяются методы GET/POST/PUT/DELETE на предмет IDOR, SQLi, XSS, CSRF. В случае GraphQL особое внимание уделяется структуре запросов и возможности обхода авторизации (query batching, introspection, object injection).

Этика и юридические аспекты (Permission, Bug Bounty, Disclosure) Проведение тестирования без разрешения является нарушением закона. Даже в рамках внутреннего пентеста необходимо документальное согласование. Этические практики включают ответственное раскрытие уязвимостей (Responsible Disclosure), участие в Bug Bounty-

программах и соблюдение NDA. Законодательство в разных странах может различаться, что важно учитывать при проведении тестирования трансграничных сервисов.

ГЛАВА 5. Анализ эффективности методик

Преимущества и недостатки автоматизированного vs ручного тестирования Автоматизированные методы (DAST, SAST, сканеры уязвимостей) обеспечивают высокую скорость проверки и удобны для интеграции в CI/CD, однако часто страдают от ложных срабатываний. Ручное тестирование, в частности пентест и code review, позволяет более глубоко анализировать сложные уязвимости и логические ошибки, но требует высокой квалификации и значительных затрат времени.

Ложные срабатывания и пропущенные уязвимости Одна из главных проблем автоматизации — ложные срабатывания и неполнота покрытия. Некоторые уязвимости, особенно логические и многошаговые, остаются незамеченными без вмешательства специалиста. При этом ручной анализ может быть субъективен и пропустить уязвимости из-за человеческого фактора.

Метрики оценки безопасности (CVSS, Risk Rating) Для оценки уязвимостей применяется система CVSS (Common Vulnerability Scoring System), позволяющая классифицировать уязвимости по критичности. Дополнительно могут использоваться внутренние метрики, учитывающие контекст угроз, вероятность эксплуатации и потенциальный ущерб. Использование формализованных шкал позволяет унифицировать отчеты и приоритизировать устранение проблем.

ГЛАВА 6. Тенденции и перспективы развития

AI и ML в тестировании безопасности Системы на базе искусственного интеллекта и машинного обучения активно внедряются в сферу тестирования защищенности. Они применяются для анализа больших объемов логов, предиктивного выявления аномалий, автоматической классификации уязвимостей и повышения точности обнаружения сложных атак.

DevSecOps и Shift Left Security Интеграция безопасности в ранние этапы жизненного цикла разработки (Shift Left) становится нормой. Подход DevSecOps предусматривает автоматическое выполнение тестов безопасности на этапе написания и сборки кода, что снижает затраты на устранение уязвимостей в будущем и обеспечивает постоянный контроль.

Угрозы будущего (API Security, Cloud-Native Threats) С увеличением числа облачных решений и микросервисных архитектур, внимание смещается к защите API, контейнеров и Kubernetes-сред. Современные атаки часто ориентированы на слабые конфигурации и недостаточную сегментацию в облаке. Также растёт актуальность защиты CI/CD-пайплайнов и цепочек поставки ПО.

ВЫВОД

Анализ методик тестирования защищенности веб-приложений показывает, что ни один подход не является универсальным. Статический и динамический анализ обеспечивают широкий охват типовых уязвимостей, но требуют дополнения ручным тестированием для выявления сложных логических ошибок. Современные инструменты и стандарты, такие как OWASP Testing Guide, NIST SP 800-115 и ISO/IEC 27034, формируют основу для системного и воспроизводимого подхода к обеспечению безопасности.

Для повышения защищенности веб-приложений рекомендуется использовать комбинированную стратегию, объединяющую автоматизацию, ручной анализ и регулярное обновление используемых методик. Комплексный подход позволяет учитывать как технические, так и организационные аспекты безопасности, что особенно важно в условиях постоянно развивающегося ландшафта киберугроз.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ISO/IEC 27034-1:2011. Information technology — Security techniques — Application security — Part 1: Overview and concepts. – Geneva: ISO, 2011.
2. ISO/IEC 27001:2022. Information technology — Security techniques — Information security management systems — Requirements. – Geneva: ISO, 2022.
3. Verizon. Data Breach Investigations Report 2023. – [Электронный ресурс]. – Режим доступа: https://www.verizon.com/business/resources/reports/dbir/ – Дата обращения: 10.05.2025.
4. MITRE Corporation. ATT&CK Framework. – [Электронный ресурс]. – Режим доступа: https://attack.mitre.org/ – Дата обращения: 10.05.2025.
5. Positive Technologies. Исследование угроз информационной безопасности 2023. – [Электронный ресурс]. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/ – Дата обращения: 10.05.2025.
6. OWASP. OWASP Top 10 – 2023. – [Электронный ресурс]. – Режим доступа: https://owasp.org/www-project-top-ten/ – Дата обращения: 10.05.2025.
7. Шевченко Д. А. Тестирование защищенности веб-приложений: учебное пособие. – М.: БХВ-Петербург, 2021. – 256 с.
8. Юрченко В. П., Мельников А. А. Безопасность веб-приложений. Практическое руководство. – М.: ДМК Пресс, 2020. – 384 с.